REMARKS

This application has been carefully considered in connection with the Advisory Action dated December 11, 2008. Reconsideration and allowance are respectfully requested in view of the following.

Summary of Rejections

Claims 1-25 were pending at the time of the Office Action.

Claims 1-25 were rejected under 35 USC § 102.

Summary of Response

Claims 1, 7, 15 and 19 are currently amended.

Claim 6 has been canceled herein.

Claims 5, 16, 18 and 24 were previously presented.

Claims 2-4, 8-14, 17, 20-23 and 25 remain as originally submitted.

Remarks and Arguments are provided below.

Summary of Claims Pending

Claims 1-5 and 7-25 are currently pending following this response.

Response to Rejections

Bowman-Amuah does not disclose streaming data conversion that extracts data from a first system, converts the data into a format compatible with a second system, and loads the converted data into the second system during normal operation of the first and second system. Bowman-Amuah also does not teach that the extractor, translator and loader operate in parallel. Streaming data conversion allows a data store of customer data to be converted from a legacy billing system to a new billing system with minimal system and customer outages. Rather than running sequentially as in a conventional batch process, the streaming data conversion processes disclosed in the pending application performs the extracting, converting, and loading of data generally in parallel. Typically, a batch conversion can require as many as 30 to 40 hours to complete. Access to the data stores holding the customer data is typically denied during the conversion process so that the data does not change during the process. If a customer requests access to billing data during the outage, it might not be possible to fulfill the request. Compared with the data set managed by batch conversion, the streaming data conversion blocks access to only a small number of units of data during conversion, thus requests for other units of data can be fulfilled during conversion. Further, the smaller set of data managed at any one time by streaming conversion creates a smaller demand on computing systems. The computing load is spread out over a longer period compared to batch conversion and system spikes are prevented. Through the use of a continuous

streaming process in which one work unit at a time is converted over a period of less than one minute each, the need for outages is reduced and the chances of a customer requesting billing information while the conversion is in progress are reduced.

The pending application discloses systems and methods for streaming conversion of data. The streaming conversion methods can be used to convert customer billing data stored in a legacy system format to a new billing system format and load the converted data in the new billing system format in real-time. In particular, a list of accounts to be converted is loaded into a scheduler in a conversion database, and the scheduler consults a set of scheduling rules to determine how the accounts are to be scheduled. A master controller notifies a transaction processing hub (TPH) that an account is in the process of being converted. The transaction processing hub notifies other systems of the start of the account conversion and imposes data point blocking for the account. To prevent modification of the content of the account during the conversion, access to content of the account can be temporarily blocked to processes or services that have the capability of modifying the account. This blocking of access to data during the conversion process can be referred to as data point blocking. When the conversion of an account is complete, data point blocking can be removed and services such as these can again be accessed. Viewing of account information is typically allowed even when data point blocking is in place. An extractor obtains data from the legacy billing system by sending requests to the legacy system via the transaction processing hub. Upon receiving data

from the legacy system, the extractor delivers the data to a translator, and the translator translates the legacy data into data that is in the format of the new billing system. In some embodiments, a translator that can be referred to as T1 translates the legacy data into an intermediate file format. T1 delivers the translated data to a translator that can be referred to as T2, and T2 translates the data from the intermediate file format into the format of the new billing system. Finally, a loader inserts the translated data into the new billing system, and the loader informs the transaction processing hub to close the account in the legacy billing system.

Bowman-Amuah is directed to a system, method and article of manufacture for translating an object attribute to and from a database value. (Bowman-Amuah Title). In general, Bowman-Amuah discloses a development architecture framework for constructing and maintaining application software. (Bowman-Amuah Col. 2, lines 30-65). Bowman-Amuah generally discloses a development architecture framework useful for software application batch processing, and certain implementations, such as object-oriented-type programming for converting object attributes to database values. (Bowman-Amuah Col. 192, lines 47-59). However, Bowman-Amuah does not disclose, teach or suggest streaming data conversion that extracts data from a first system, converts the data into a format compatible with a second system, and loads the converted data into the second system during normal operation of the first and second

system, and performs the extracting, converting and loading generally in parallel, as claimed.

These distinctions, as well as others, will be discussed in greater detail in the analysis of the present claims that follows.

Response to Rejections under Section 102

Claim 1:

Claim 1 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah, U.S. Patent No. 6,529,909 ("Bowman-Amuah").

I. Bowman-Amuah also does not disclose that the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the normal operation comprises operating on data from the first system other than the unit of data from the first system during the conversion of the unit of data from the first system.

Claim 1 has been amended to clarify normal operation of first and second systems. Claim 1 now specifically recites, "the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the normal operation comprises operating on data from the first system other than the unit of data from the first system during the conversion of the unit of data from the first system." Applicants submit that no new matter has been added, and support for the amendment maybe found at least in paragraph [0014] and [0024].

Bowman-Amuah's framework is intended to provide an inventory of software components needed to design, build, install and operate the systems involved, and an understanding of how the components should fit together conceptually. (Bowman-Amuah Col. 20, lines 14-20). Bowman-Amuah describes the benefits of its architectural approach to software development as increased productivity and "less reinvention of the wheel." (Bowman-Amuah Col. 20, lines 45-53). As a specific implementation of its architectural approach to software development, Bowman-Amuah discloses a method for translating an object attribute to and from a database value. Bowman-Amuah describes its method as a type of object-oriented programming and defines an "object" as a self-sufficient software package that contains both data and a collection of related structures and procedures, and does not require other additional structures, procedures or data to perform its specific task. (Bowman-Amuah Col. 192, lines 47-59). The method determines a conversion process to be used for converting the object attribute to and from a database value. The conversion process is then encapsulated in an attribute converter. The object attribute is then directed to the attribute converter to be converted to a database value.

While Bowman-Amuah may disclose a conversion process, Applicants submit that Bowman-Amuah does not teach or suggest streaming conversion of data from a first format to a second format. Compared to a conventional batch process, the streaming data conversion processes disclosed in the pending application performs the

extracting, converting and loading of data generally in parallel. As disclosed in paragraph 0006 of the pending application, a batch conversion can typically require as many as 30 to 40 hours to complete and access to the data stores holding the customer data is typically denied during the conversion process so that the data does not change during the process. If a customer requests access to billing data during the outage, it might not be possible to fulfill the request. However, as disclosed in paragraphs 0026 and 0045 of the pending application, the streaming data conversion blocks access to only a small number of units of data during conversion, thus requests for other units of data can be fulfilled during conversion. Accordingly, Applicants submit that Bowman-Amuh does not teach or suggest that the extractor, the translator, and loader components convert the unit of data during normal operation of the first and second systems, wherein the normal operation comprises at least one other system operating on data other than the unit of data from the first system during the conversion of the unit of data from the first system, as claimed.

II. Bowman-Amuah does not disclose an extractor component that extracts a unit of data from the first system.

Claim 1 recites, "an extractor component that extracts a unit of data from the first system."

The Final Office Action dated August 20, 2008 relied on the following disclosure in Bowman-Amuah (Col. 20, lines 25-34) to read on the extractor component recited in claim 1:

> Frameworks are used to help practitioners understand what components may be required and how the components fit together. Based on the inventory of components and the description of their relationships, practitioners will select the necessary components for their design. An architect extracts components from one or more Frameworks to meet a specific set of user or application requirements. Once an architecture has been implemented it is often referred to as an architecture or an infrastructure. (Underlining added for emphasis.)

As shown above, Bowman-Amuah merely discloses one or more steps that an architect might take to meet a set of user or application requirements. As shown, the word "extracts" is disclosed. However, Bowman-Amuah does not disclose what components would be extracted or how such an extraction might actually be accomplished (e.g., manually, computer-assisted, etc.). Clearly, Bowman-Amuah's disclosure of an architect extracting components from one or more frameworks to meet user or application requirements does not teach or suggest "an extractor component that extracts a unit of data from the first system," as recited in claim 1.

III. Bowman-Amuah also does not disclose that the extractor, translator and loader components extract, convert and load generally in parallel.

Claim 1 recites, "wherein the extractor, translator and loader components extract, convert and load generally in parallel."

The Advisory Action dated December 11, 2008 relied on Bowman-Amuah's Col. 117, lines 60-65, and Col. 197, lines 60-67 to teach that the extractor, translator and loader components extract, convert, and load data generally in parallel. Applicants note that Bowman-Amuah provides disclosure of the Route Management of the components of the Workflow in Col. 117, lines 60-65 reproduced below:

> Route management enables the routing of tasks to the next role, which can be done in the following ways:
> Serial---the tasks are sequentially perfomed;
> Parallel---the work is divided among different players;

Thus, Applicants submit that Bowman-Amuah teaches that route management is part of a Workflow which enables tasks within a business process to be passed among the appropriate participants. While Bowman-Amuah may teach parallely dividing tasks among different players, Bowman-Amuah does not teach data conversion by extracting, translating, and loading in parallel. Further, in Col. 197, lines 60-67 Bowman-Amuah teaches benefits of encapsulating each processing step in a batch processing system within a filter component. In particular, Bowman-Amuah teaches that each filter performs its data processing independently of other filters, and there may be several instances of a particular type of filter running in parallel. Thus, Applicants submit that Bowman-Amuah teaches a batch processing system comprising filter components.

Applicants submit that Bowan-Amuah does not teach streaming conversion of data by extracting, translating and loading in parallel, as claimed.

Moreover, the Advisory Action states that "wherein the extractor, translator and loader components extract, convert and load generally in parallel' implies that at times the system is not operating in parallel." Applicants respectfully traverse. Based on the logic of the Advisory Action, the claim language equally implies that at times the system **is** operating in parallel. Further, even if hypothetically, claim 1 implies precisely as suggested by the Advisory Action (which Applicants do not admit to), Applicants submit that based on the discussion of Bowman-Amuah above, Bowman-Amuah still fails to teach or suggest that the extractor, translator, and loader components extract, convert, and load data generally in parallel, as claimed

The Final Office Action dated August 20, 2008 relied on the following disclosure in Bowman-Amuah (Col. 193, line 63 to Col. 194, line 8) to read on the loader component recited in claim 1:

> Abstraction Factory:
>
> AbstractType produceForKey(key)
>
> Abstract Type:
>
> init(some data stream)
>
> the Abstraction Factory can be fully coded in C++. It is very re-usable as it stands. In addition, it has been extended to perform 'Java Loader-like' dynamic linking if the proper code cannot be found already within the factory.
>
> Factory, the well know[sic] pattern from Gamma, et. al BUW, in which the objects created by the factory can be dealt with

> generically in terms of independence, scalability, parallel processing, etc. Component Solutions Handbook.

As shown above, Bowman-Amuah describes the use of an Abstraction Factory in an object-oriented programming environment. The above-disclosed section of Bowman-Amuah clearly does not teach or suggest extractor, translator, and loader components that extract, convert, and load data generally in parallel, as recited in claim 1.

In Col. 192, lines 46-65, Bowman-Amuah describes the details of an Abstraction Factory. In particular, Bowman-Amuah discloses a method for providing an abstraction factory pattern. According to the method disclosed, data is received and transformed into a plurality of concrete objects. Each of the concrete objects is associated with an abstract interface. A map of the association between the concrete objects and the abstract interface is then created. Thus, when a request for an abstraction pattern is received, the request includes an identifier for one of the concrete objects and an identifier for the abstract interface. The map is then consulted to locate the concrete object that has been identified. An abstract object is then created that corresponds to the located concrete object.

Additionally, in Col. 193, lines 20-43, Bowman-Amuah further describes its Abstraction Factory. Specifically, Bowman-Amuah discloses:

> ...one transforms the various types of raw data into a corresponding variety of concrete object types, all of which share a common abstract interface. This transformation will

> be encapsulated within an Abstraction Factory. The primary interface to the Abstraction Factory is:
>
> 'abstractType produceForKey(key)'
>
> where 'abstractType' is the type of the common abstract interface, and key is a piece of information which identifies the appropriate concrete type. ...When this method is invoked, the Abstraction Factory consults its internal mapping and creates an 'empty' object of the proper concrete class. The factory then casts the concrete object into the abstraction and returns it to the method's client. This client (a framework most likely) will then instruct the abstraction to initialize itself from the incoming data stream.
>
> At the end of this process we have an abstract handle to a concrete object which a framework may then manipulate generically.

As shown by all of the above, Bowman-Amuah's disclosure of its "Abstraction Factory" does not teach or suggest that the extractor, translator, and loader components extract, convert, and load data generally in parallel, as recited in claim 1.

Therefore, for at least the reasons established above in sections I, II, and III, Applicants respectfully submit that independent claim 1 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 1:

Claims 2-6 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent claim 6 has been canceled herein. Dependent claims 2-5 depend directly or indirectly from independent claim 1 and incorporate all of the limitations

thereof. Accordingly, for at least the reasons established in sections I, II, and III above, Applicants respectfully submit that claims 2-5 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Claim 7:

Claim 7 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Claim 7 includes limitations substantially similar to the limitations discussed in sections I, II, and III above. For example, claim 7 recites, "extracting a unit of data from a database associated with the first system; ... normally accessing data other than the unit of data from the first and second systems, wherein normally accessing comprises at least one other system accessing data other than the unit of data during the extraction, translation, and loading of the unit of data and, wherein the extracting, translating and loading are performed generally in parallel." Therefore, for at least the reasons established above in sections I, II, and III, Applicants respectfully submit that independent claim 7 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 7:

Claims 8-14 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent claims 8-14 depend directly or indirectly from independent claim 7 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I, II, and III above, Applicants respectfully submit that claims 8-14 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Claim 15:

Claim 15 was rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Claim 15 includes limitations substantially similar to the limitations discussed in sections I, II, and III above. For example, claim 15 recites, "an extractor component that extracts a unit of data from the first system; ... the extractor, the translator, and the loader components extract, convert, and load the unit of data during normal operation of the first and second systems, wherein the normal operation comprises at least one other system operating on a second portion of the data of the first system during the conversion of the unit of data from the first system, and wherein the extractor, translator and loader components extract, translate and load generally in parallel." Therefore, for at least the reasons established above in sections I, II, and III, Applicants respectfully submit that independent claim 15 is not anticipated by Bowman-Amuah and respectfully request allowance of this claim.

Claims Depending From Claim 15:

Claims 16-25 were rejected under 35 USC § 102(e) as being anticipated by Bowman-Amuah.

Dependent claims 16-25 depend directly or indirectly from independent claim 15 and incorporate all of the limitations thereof. Accordingly, for at least the reasons established in sections I, II, and III above, Applicants respectfully submit that claims 16-25 are not anticipated by Bowman-Amuah and respectfully request allowance of these claims.

Conclusion

Applicants respectfully submit that the pending application is in condition for allowance for the reasons stated above. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, the Examiner is encouraged to telephone the undersigned at (972) 731-2288.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: January 21, 2009

/Michael W. Piper/
Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANTS

CONLEY ROSE, P.C.
5601 Granite Parkway, Suite 750
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 (facsimile)